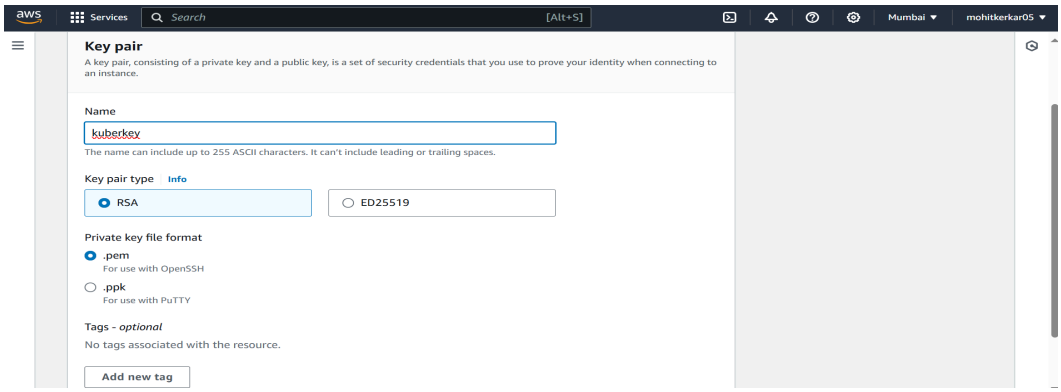
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Lab Exp 04

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application. (I performed this Experiment on my personal AWS account)

Step 1: Creation of EC2 Instance, its Key-pair, remote login using SSH

Firstly, create a key pair with a desired name and type 'RSA' with .pem extension. (.pem is useful with OpenSSH)

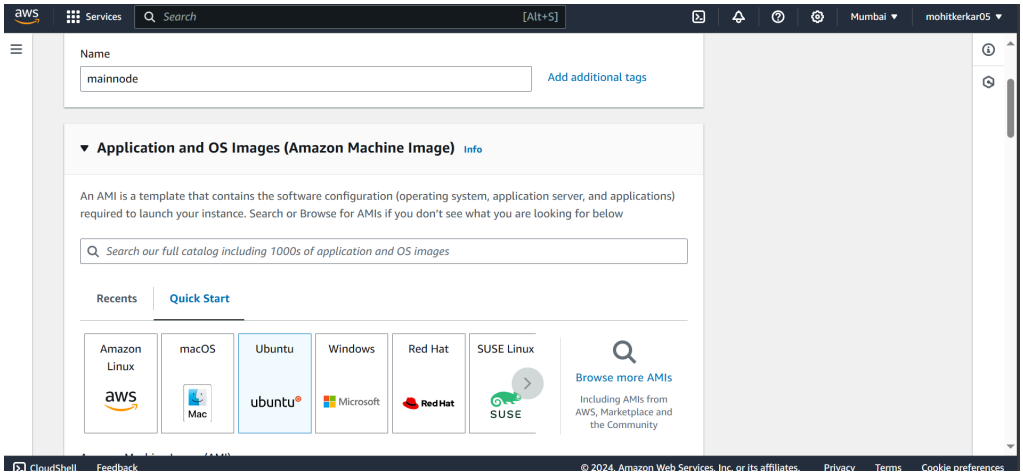


After creating the key, it automatically gets downloaded onto your machine. We are to create a new folder and place our within there. After this, we are to access our key and establish our connection with our instance by changing the directory to our key's directory

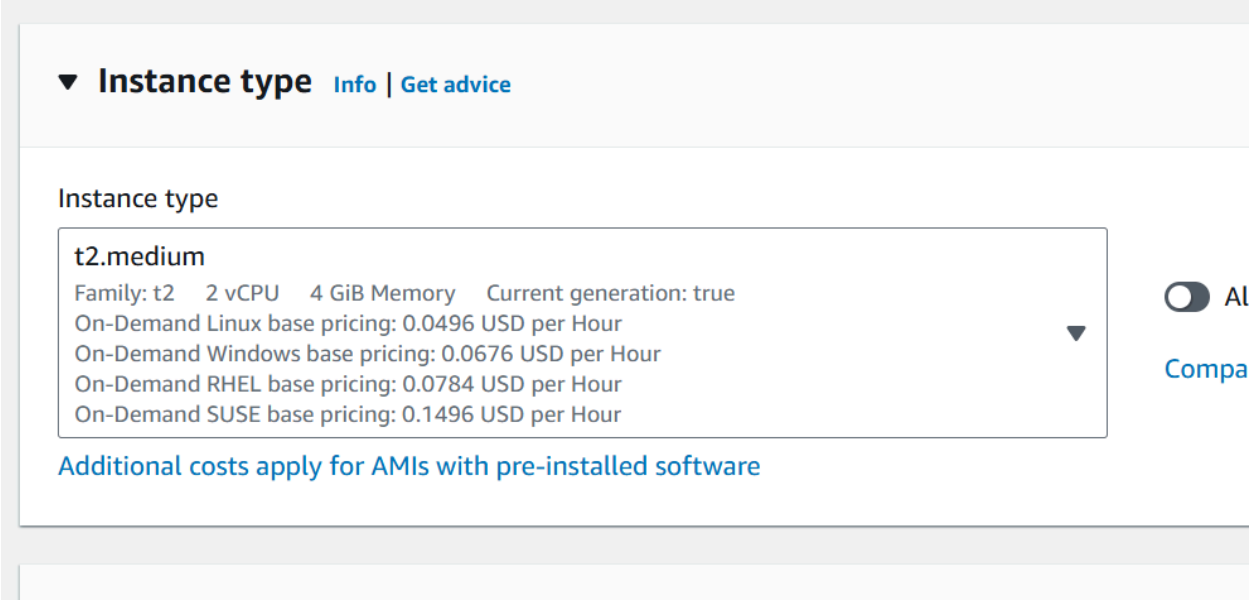
Now, proceed to the creation of the EC2 instance. I selected Amazon machine image as **Ubuntu** and instance type as **t2.medium**, just for the purpose of convenience and also for successful execution of the experiment.

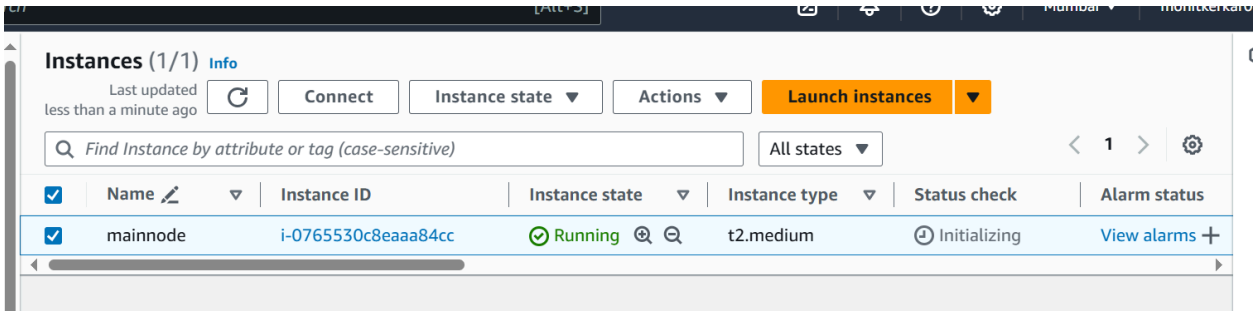
The execution of the experiment is determined by the fact that how many resources is the instance allowed to access or use. The Free Tier eligible instance type **t2.micro avails the instance with only one** CPU, whereas with t2.medium we get 2 CPU and much more execution space.

Be mindful that along with all this, **t2.medium does not have free tier eligibility**. So, use the instances and the resources that you allow them to access resourcefully and economically. Shut/terminate them as and when work related to them is over

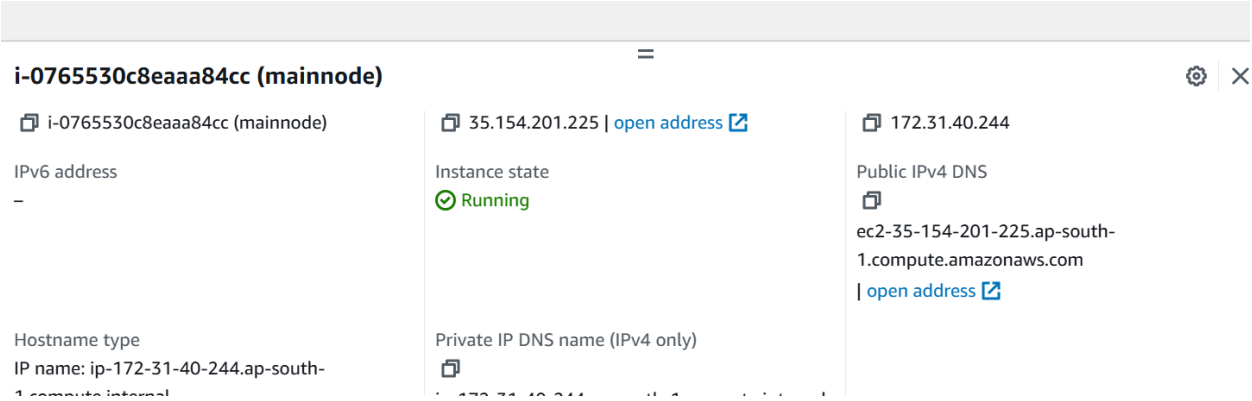


Name: Mohit Kerkar Div: D15C Roll No: 23



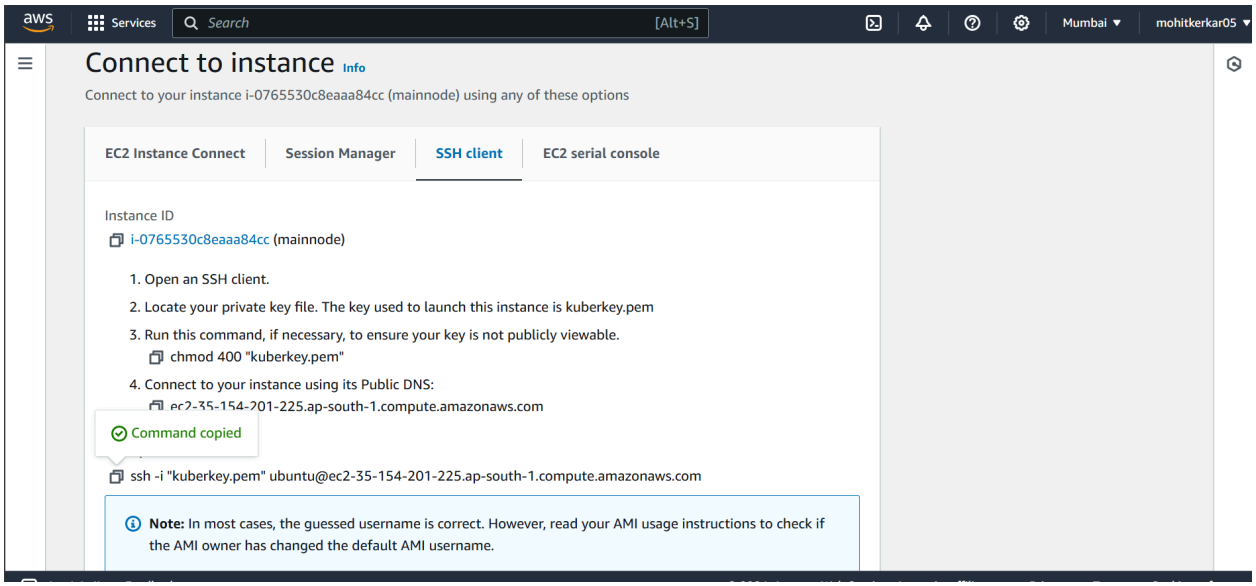


Here are my instance details. From here on, after remotely logging in from my local terminal, you would see my IPv4 address in my terminal (Powershell)

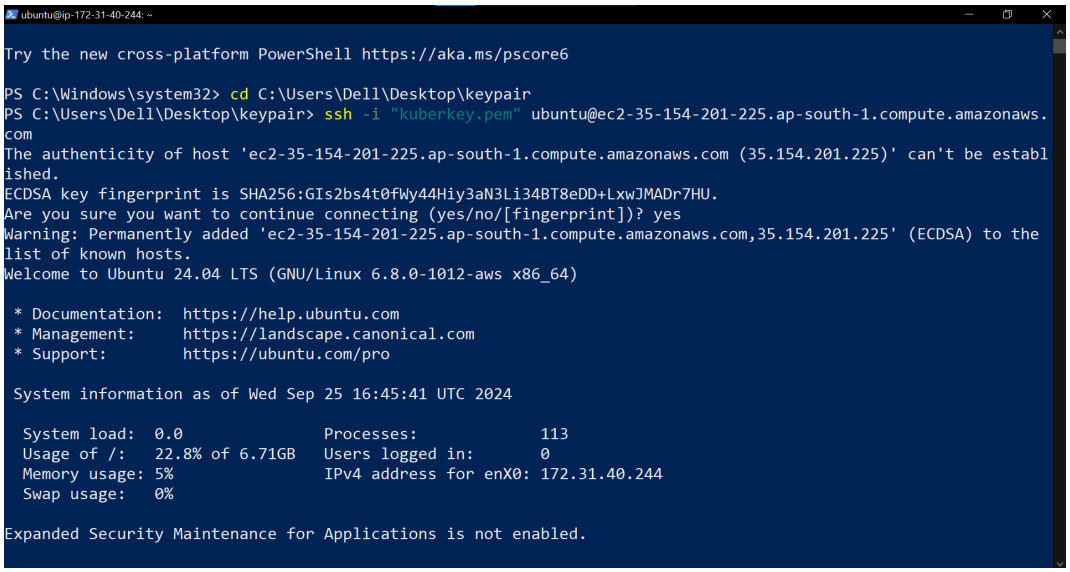


Name: Mohit Kerkar Div: D15C Roll No: 23

Select the instance and press connect. Navigate to the SSH client section and copy the SSH command provided to us. We will require this for remotely logging into our instance from our local terminal.



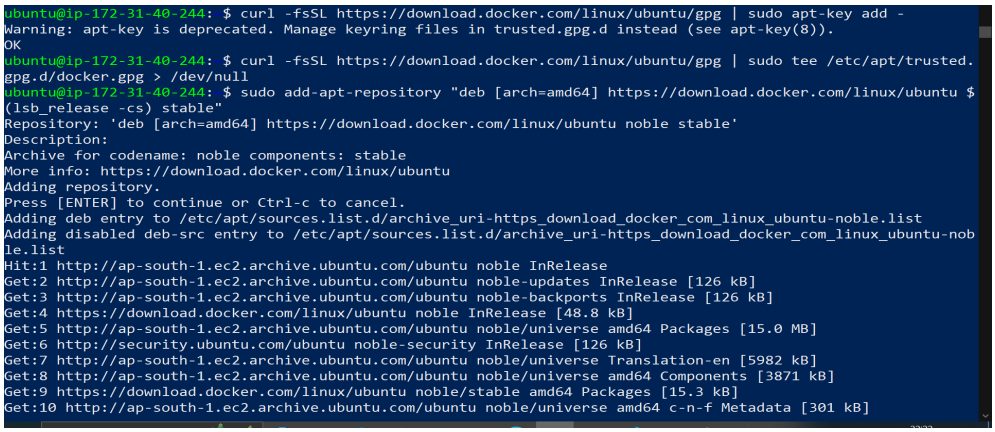
As one can see, I changed my directory to the folder in which i had stored my key (.pem) and ran the ssh command so as to remotely login onto my instance and perform the necessary tasks from my terminal itself



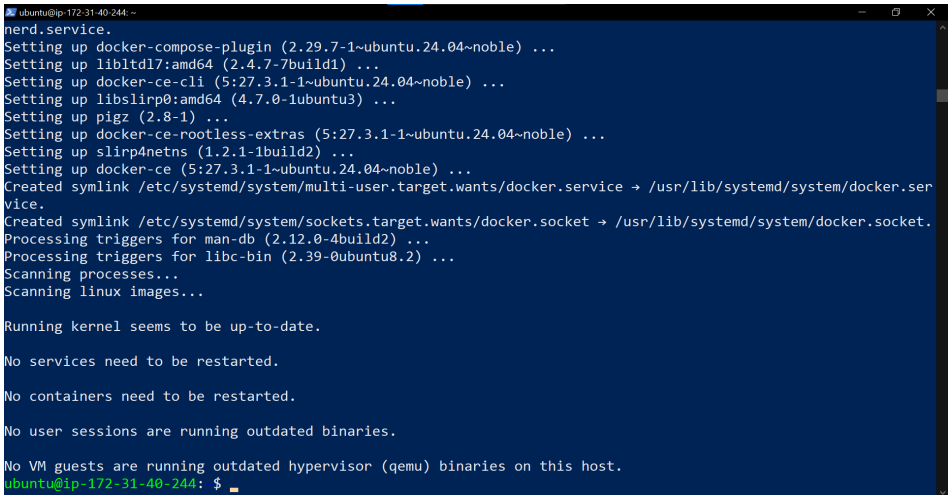
Step 2: Docker installation

Run the below mentioned commands in order to install docker on your instance

Adding key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Adding key without apt-key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
Add docker repository: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



Run these commands to update the package list and install Docker:
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io



Next, run this command….
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-40-244:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-40-244:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-244:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart docker
```

Step 3: Kubernetes Installation

Run these commands to install kubernetes:
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-40-244:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dear
mor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-40-244:~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
io/core:/stable:/v1.31/deb/ /" | Out-File -FilePath "C:\path\to\your\directory\kubernetes.list" -Encoding utf8
Out-File: command not found
ubuntu@ip-172-31-40-244:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-40-244:~$
```

After this step, run the following commands,
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

Over here, while running sudo apt-get update if you encounter an error like this

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
```

Then run the following command and ensure that the file contains the following texts:
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /

```
ubuntu@ip-172-31-40-244: ~
  GNU nano 7.2                    /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

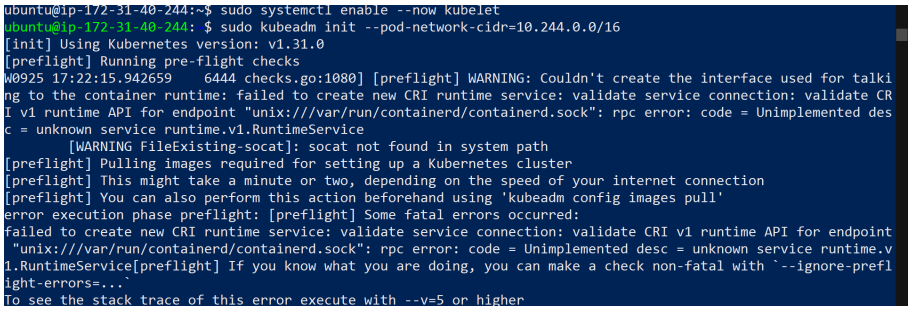
Be sure to save this file and then proceed with these commands

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

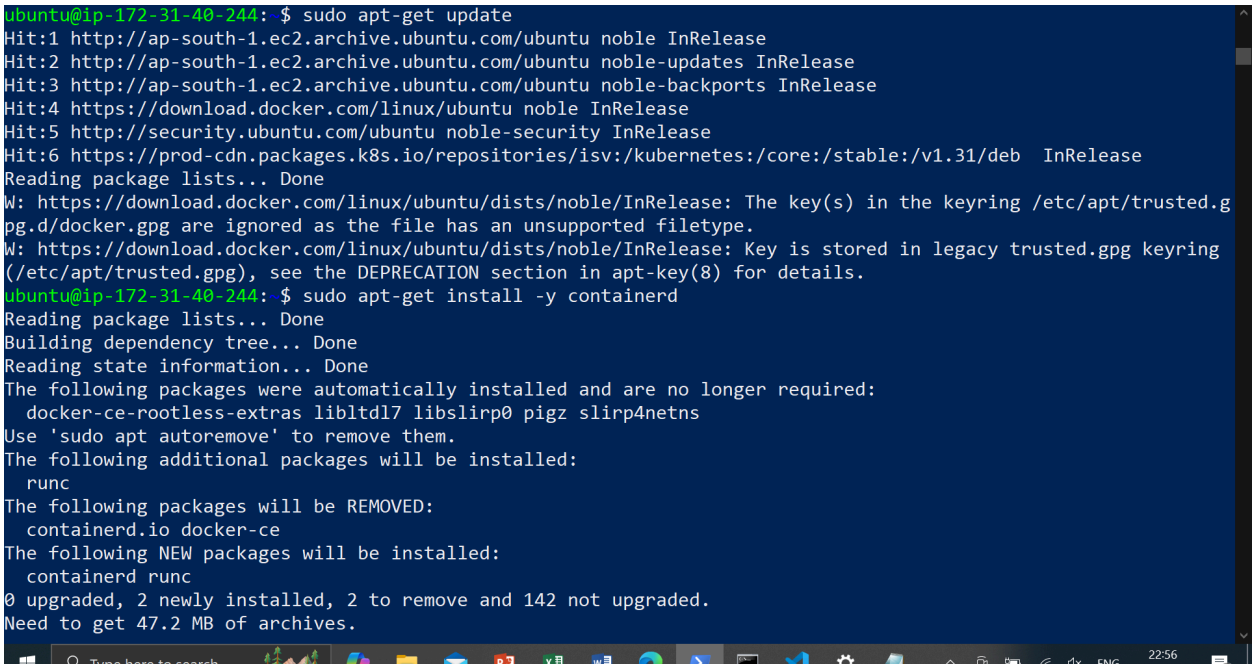
```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B
]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 132 kB in 1s (199 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.g
pg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
(/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 k
```

```
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-40-244:~$
```

sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16



Since running that command ran us into an error, we are expected to install containerd first
Run this command: sudo apt-get install -y containerd
Then run,
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml



Post this step, run these commands one by one
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

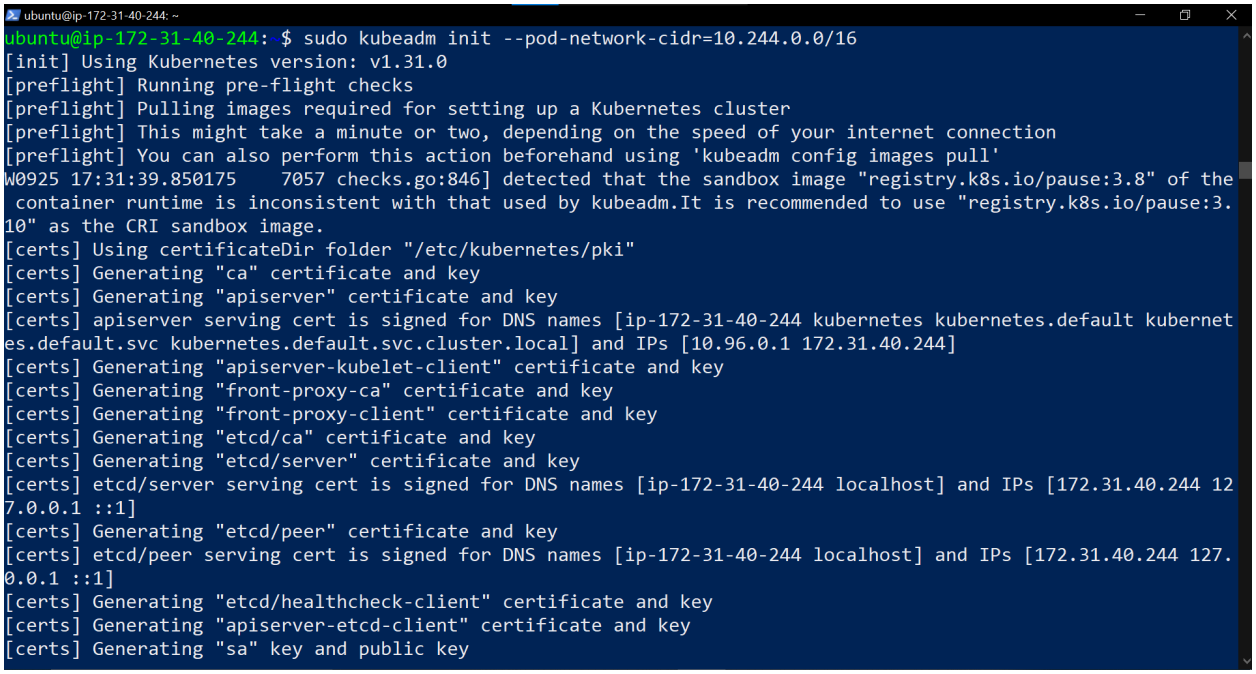
```
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-25 17:23:54 UTC; 17s ago
       Docs: https://containerd.io
   Main PID: 6555 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 14.1M)
        CPU: 121ms
     CGroup: /system.slice/containerd.service
             └─6555 /usr/bin/containerd

Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231856399Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231887455Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231896683Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231919602Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231952908Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231972572Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231980403Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231986601Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 systemd[1]: Started containerd.service - containerd container runtime.
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.234185443Z" level=info msg="conta>
```

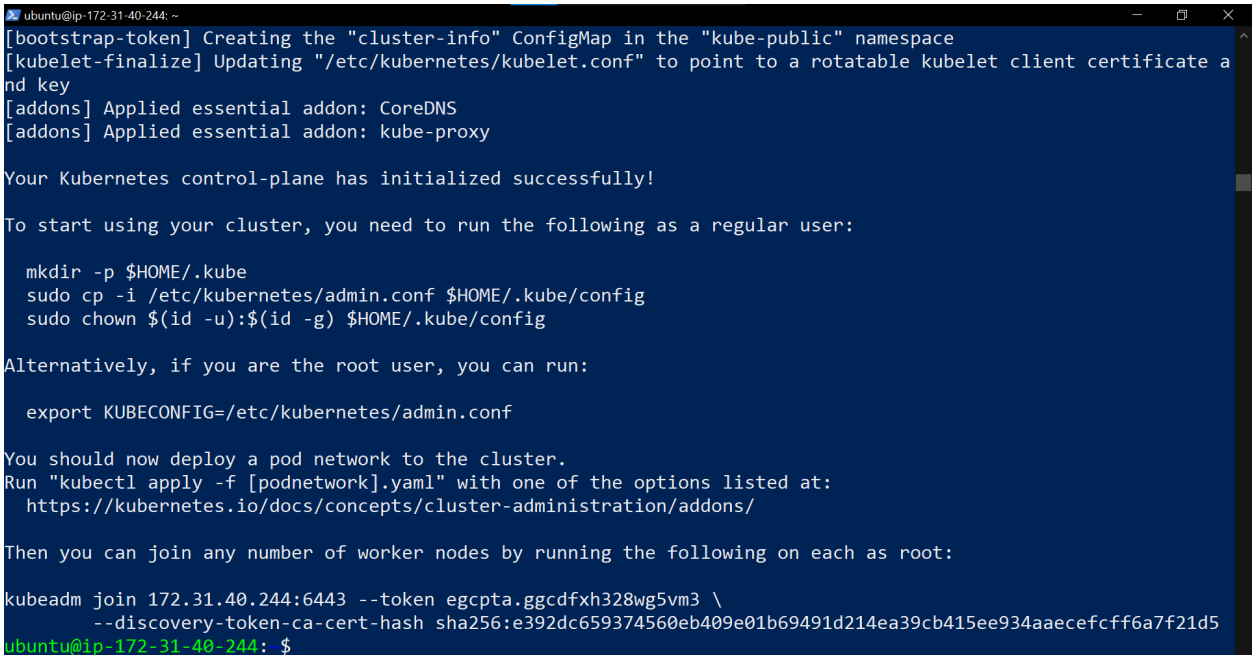
After this, run
sudo apt-get install -y socat

```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

Step 4: Initialize kubecluster:
Sudo kubeadm init –pod-network-cidr=10.244.0.0/16

Name: Mohit Kerkar Div: D15C Roll No: 23





Copy the mkdir and chown commands from the top and execute them.

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentati
on/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-40-244:~$
```

kubectl get pods

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7    0/1     Pending   0          48s
nginx-deployment-d556bf558-s5rtj    0/1     Pending   0          48s
ubuntu@ip-172-31-40-244:~$
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-40-244:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-40-244:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-40-244:~$
```

To untaint all nodes, run the following command
kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

Then run, kubectl get nodes

```
ubuntu@ip-172-31-40-244:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-40-244 untainted
ubuntu@ip-172-31-40-244:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-40-244   Ready    control-plane   16m   v1.31.1
ubuntu@ip-172-31-40-244:~$
```

Then run, kubectl get pods command again

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7    1/1     Running   0          7m15s
nginx-deployment-d556bf558-s5rtj    1/1     Running   0          7m15s
ubuntu@ip-172-31-40-244:~$
```

Since, Jenkins server/service is already occupying port 8080, i tried starting the nginx service at 8082 port number. Following is the command that i executed was,
kubectl port-forward $POD_NAME 8082:80

```
ubuntu@ip-172-31-40-244:~$ kubectl port-forward svc/nginx 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
```

The last step consisted of opening a new terminal and logging in using the SSH method that we used earlier and running the following command so as to finally confirm that we have successfully deployed our kubernetes application over the nginx server, using nginx service (opional)

curl --head http://127.0.0.1:8082

```
Last login: Wed Sep 25 19:13:52 2024 from 45.112.58.76
ubuntu@ip-172-31-40-244:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 19:36:26 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

Conclusion:
In this experiment, we successfully deployed an Nginx web server on a Kubernetes cluster and exposed it using NodePort services. By creating the deployment and configuring the service, we demonstrated Kubernetes' ability to manage containerized applications efficiently.
The use of `kubectl` commands allowed us to check pod status and access the Nginx server locally via port forwarding on the EC2 instance. This experience provided valuable insights into service management and networking concepts within Kubernetes.
Overall, this experiment reinforced the fundamental skills needed for deploying cloud-native applications and laid the groundwork for exploring more advanced Kubernetes features in future projects.